arXiv:2302.02801v1 [cs.LG] 3 Feb 2023

LAMPP: Language Models as Probabilistic Priors for Perception and Action

Belinda Z. Li [1] **William Chen** [1] **Pratyusha Sharma** [1] **Jacob Andreas** [1]

Abstract

Language models trained on large text corpora encode rich distributional information about real-world environments and action sequences. This information plays a crucial role in current approaches to language processing tasks like question answering and instruction generation. We describe how to leverage language models for *non-linguistic* perception and control tasks. Our approach casts labeling and decision-making as inference in probabilistic graphical models in which language models parameterize prior distributions over labels, decisions and parameters, making it possible to integrate uncertain observations and incomplete background knowledge in a principled way. Applied to semantic segmentation, household navigation, and activity recognition tasks, this approach improves predictions on rare, out-of-distribution, and structurally novel inputs.

1. Introduction

Common-sense priors are crucial for decision-making under uncertainty in real-world environments. Suppose that we wish to label the objects in the scene depicted in Fig. 1(b). Once a few prominent objects (like the bathtub) have been identified, it is clear that the picture depicts a bathroom. This helps resolve some more challenging object labels: the curtain in the scene is a shower curtain, not a window curtain; the object on the wall is a mirror, not a picture. Prior knowledge about likely object or event co-occurrences are essential not just in vision tasks, but also for navigating unfamiliar places and understanding other agents' behaviors. Indeed, such expectations play a key role in human reasoning for tasks like object classification and written text interpretation (Kveraga et al., 2007; Mirault et al., 2018).

In most problem domains, current machine learning models acquire information about the prior distribution of labels and decisions from task-specific datasets. Especially when training data is sparse or biased, this can result in systematic errors, particularly on unusual or out-of-distribution inputs. How might we endow models with more general and flexible prior knowledge?

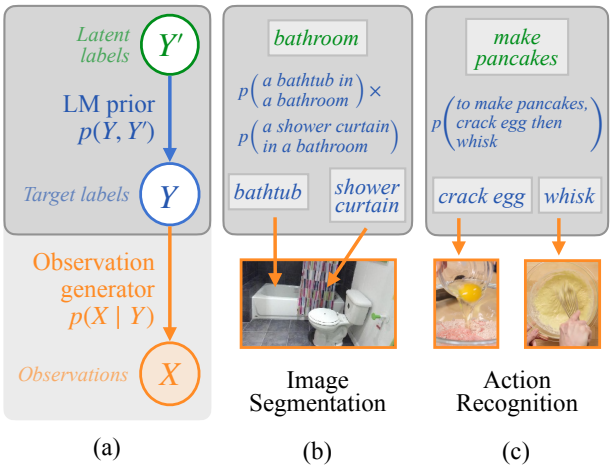

Figure 1. In LAMPP, the LM provides a prior over a structured label space $P(Y, Y')$ and a task-specific observation model provides $P(X \mid Y)$. We apply LAMPP to three concrete tasks, including image segmentation and video action recognition.* In the image segmentation case, the LM provides a prior over what objects are likely to co-occur (based on room-object probabilities), which allows it to determine that the observed curtain is a *shower curtain*. In the action recognition case, the LM provides a prior over what action sequences are likely to accomplish the target tasks, allowing it to infer the action sequence in a video.

*Our third task, object navigation, is not shown in this figure.

We propose to use **language models**—learned distributions over natural language strings—as task-general probabilistic priors. Unlike segmented images or robot demonstrations, large text corpora are readily available and describe almost all facets of human experience. Language models (LMs) trained on them encode much of this information—like the fact that *plates are located in kitchens and dining rooms*, and that *whisking eggs is preceded by breaking them*—often with greater diversity and fidelity than can be provided by small, task-specific datasets. Such linguistic supervision has also been hypothesized to play a role in aspects of human common-sense knowledge that are difficult to learn from direct experience (Painter, 2005).

In language processing and other text generation tasks, LMs

[1]MIT CSAIL, Cambridge, Massachusetts, USA. Correspondence to: Belinda Z. Li <bzl@mit.edu>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

have been used as sources of prior knowledge for tasks spanning common-sense question answering (Talmor et al., 2021), modeling scripts and stories (Ammanabrolu et al., 2020; 2021), and synthesis of probabilistic programs (Lew et al., 2020). They have also been applied to grounded language understanding problems via **model chaining** (MC) approaches, which encode the output of perceptual systems as natural language strings that prompt LMs to directly generate labels or plans (Zeng et al., 2023; Singh et al., 2022).

In this paper, we instead focus on LMs as a source of probabilistic background knowledge that can be integrated with existing domain models. LMs pair naturally with structured probabilistic modeling frameworks: by using them to place *prior* distributions over labels, decisions or model parameters, we can combine them with domain-specific generative models or likelihood functions to integrate "top-down" background knowledge with "bottom-up" task-specific predictors. This approach offers a principled way to integrate linguistic supervision with structured uncertainty about non-linguistic variables, making it possible to leverage LMs' *knowledge* even in complex tasks where LMs struggle with *inference*.

We call this approach to modeling **LAMPP** (**La**nguage **M**odels as **P**robabilistic **P**riors). LAMPP is flexible and applicable to a wide variety of problems. We present three case studies featuring tasks with diverse objectives and input modalities—semantic image segmentation, robot navigation, and video action segmentation. LAMPP consistently improves performance on rare, out-of-distribution, and structurally novel inputs, and sometimes even improves accuracy on examples within the domain model's training distribution. These results show that language is a useful source of background knowledge for general decision-making, and that uncertainty in this background knowledge can be effectively integrated with uncertainty in non-linguistic problem domains.

2. Method

A language model (LM) is a distribution over natural language strings. LMs trained on sufficiently large text datasets become good models not just of grammatical phenomena, but various kinds of world knowledge (Talmor et al., 2021; Li et al., 2021). Our work proposes a method for extracting probabilistic common-sense priors from language models, which can then be used to supplement and inform *arbitrary* task-specific models operating over multiple modalities. These priors can be leveraged at multiple stages in the machine learning pipeline:

Prediction: In many learning problems, our ultimate goal is to model a distribution $p(y \mid x)$ over labels or decisions y given (non-linguistic) observations x. These ys might be structured objects: in Fig. 1(b), x is an image and y is a set of labels for objects in the image. By Bayes' rule, we can write:

$$p(y \mid x) \propto p(y)p(x \mid y) \,, \tag{1}$$

which factors this decision-making problem into two parts: a prior over labels $p(y)$, and a generative model of observations $p(x \mid y)$. If we have such a generative model, we may immediately combine it with a representation of the prior $p(y)$ to model the distribution over labels.

Learning: In models with interpretable parameters, we may also leverage knowledge about the distribution of these parameters themselves during learning, before we make any predictions at all. Given a dataset $\mathcal{D}$ of examples (x_i, y_i) and a predictive model $p(y \mid x; \theta)$, we may write

$$\begin{aligned} p(\theta \mid \mathcal{D}) &\propto p(\mathcal{D} \mid \theta)p(\theta) \\ &= \Big(\prod_i p(y_i \mid x_i; \theta)\Big)p(\theta) \,, \end{aligned} \tag{2}$$

in this case making it possible to leverage prior knowledge of θ itself, e.g., when optimizing model parameters or performing full Bayesian inference.

In structured output spaces, like segmented images, robot trajectories, or high-dimensional parameter vectors, a useful prior contains information about which joint configurations are plausible (e.g., an image might contain sofas and chairs, or showers and sinks, but not sinks and sofas). How can we use an LM to obtain and use distributions $p(y)$ or $p(\theta)$? Applying LAMPP in a given problem domain involves four steps:

1. **Choosing a base (domain) model**: Here we can use any model of observations $p(x \mid y)$ or labels $p(y \mid x; \theta)$.

2. **Designing a label space**: When reasoning about a joint distribution over labels or parameters, correlations between these variables might be expressed most compactly in terms of some other latent variable (in Fig. 1(b), object labels are coupled by a latent *room*). Before querying an LM to obtain $p(y)$ or $p(\theta)$, we may introduce additional variables like this one to better model probabilistic relationships among labels.

3. **Querying the LM**: We then obtain scores for each configuration of y or θ by *prompting* a language model with a query about the plausibility of the configuration, then *evaluating* the probability that the LM assigns to the query. Examples are shown in Fig. 1(b–c). For all experiments in this paper, we use the GPT-3 to score queries (Brown et al., 2020).

4. **Inference**: Finally, we perform inference in the graphical model defined by $p(y)$ and $p(x \mid y)$ (or $p(\theta)\, p(y \mid x, \theta)$) to find the highest-scoring (or otherwise risk-minimizing) configuration of y for a given x.

In Sections 3–5, we apply this framework to three learning problems. In each section, we evaluate LAMPP's ability to improve *generalization* over base models. We focus on three types of generalization: **zero-shot** (ZS), **out-of-distribution** (OOD), and **in-distribution** (ID). The type of generalization required depends on the availability and distribution of training data: ZS evaluations focus on the case in which $p(x \mid y)$ is known (possibly just for components of y, e.g., appearances of individual objects), but no information about the joint distribution $p(y)$ (e.g., configurations of rooms) is available at training time. OOD evaluations focus on biased training sets (in which particular label combinations are over- or under-represented). ID evaluations focus on cases where the full evaluation distribution is known and available at training time.

3. LAMPP for Semantic Segmentation

We first study the task of **semantic image segmentation**: identifying object boundaries in an image and labeling each object x_i with its class y_i. How might background knowledge from an LM help with this task? Intuitively, it may be hard for a bottom-up visual classifier to integrate global image context and model correlations among distant objects' labels. LMs encode common-sense information about the global structure of scenes, which can be combined with easy-to-predict object labels to help with more challenging predictions.

3.1. Methods

Base model Standard models for semantic segmentation discriminatively assign a label y_i to each pixel x_i in an input image x according to some:

$$p_{\text{seg}}(y_i \mid x)\ . \tag{3}$$

Our experiments use RedNet (Jiang et al., 2018), a ResNet-50-based autoencoder model, to compute Eq. (3). By computing $\arg\max_y\ p(y \mid x)$ for each pixel in an input image, we obtain a collection of **segments**: contiguous input regions assigned the same label (see bottom of Fig. 2). When applying LAMPP, we treat these segments as given, but attempt to choose a better joint labeling for all segments in an image.

Label Space We do so using the generative model depicted in Fig. 2. We hypothesize a generative process in which every image originates in a **room** r. Conditioned on the room, a fixed number of **objects** are generated, each with label y_i. To model possible perceptual ambiguity, each true object labels in turn generates a **noisy** object label d_i. Finally, each of these generates an image segment x_i.

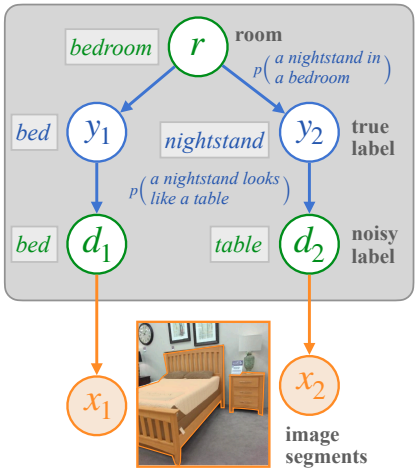

Figure 2. Generative model for **image semantic segmentation**. Images originate in a room r, which generates the objects y_1, y_2 in the room, which generate noisy object labels d_1, d_2 representing perceptually similar objects. Finally, each d_i generates an image segment x_i, a continuous region of image pixels depicting each object. Rooms r, true labels y_i, and noisy labels d_i are latent, while image segments x_i are observed.

We use the base segmentation model p_{seg} to compute $p(x_i \mid d_i)$ by applying Bayes' rule locally for each segment: $p(x_i \mid d_i) \propto p_{\text{seg}}(d_i \mid x_i)/p(d_i)$. All other distributions in this generative model are parameterized by an LM, as described below. Ultimately, we wish to recover "true" object labels y_i; the latent labels r and d help extract usable background information about objects' co-occurrence patterns and perceptual properties.

LM Queries We compute the object–room co-occurrence probabilities $p(y_i \mid r)$ by prompting the LM with the string:

A(n) [r] *has a(n)* [y_i]: [*plausible / implausible*]

The LM conditions on the non-highlighted portion of the prompt and is expected to generate one of the highlighted tokens. We compute the relative probability the LM assigns to tokens *plausible* and *implausible*, then normalize these over all object labels y to parameterize the final distribution. We use the same procedure to compute the object–object confusion model $p(d_i \mid y_i)$, prompting the LM with:

The [d_i] *looks like the* [y_i]: [*plausible / implausible*]

Inference The model in Fig. 2 defines a joint distribution over all labels $\underline{y} = y_1, \ldots, y_n$. To re-label a segmented image, we compute the max-marginal-probability label for

each segment independently:

$$\begin{aligned}&\arg\max p(y_i \mid \underline{x}) \\ &\quad = \arg\max \sum_r \sum_{\underline{y}\setminus\{y_i\}} \sum_{\underline{d}} p(\underline{x}, \underline{d}, \underline{y}, r) \qquad (4)\end{aligned}$$

The form of the decision rule used for semantic segmentation (which includes several simplifications for computational efficiency) can be found in Appendix A.1.

3.2. Experiments

We use the SUN RGB-D dataset for our semantic segmentation tasks (Song et al., 2015), which contains RGB-D images of indoor environments. We also implement a model-chaining (MC) baseline that integrates LM knowledge without considering model uncertainties. We take noisy labels from the image model (d_i) and directly query the LM for true labels (y_i). Details of this baseline can be found in Appendix A.2. We attempt to make the MC inference procedure as analogous to our approach as possible: the LM must account for both room-object co-occurrence likelihoods and object-object resemblance likelihoods when predicting true labels. However, here, the LM must implicitly incorporate these likelihoods into its text-scoring, rather than integrating them into a structured probabilistic framework. We evaluate the RedNet *base model*, this *model chaining* approach, and LAMPP on **in-distribution** and **out-of-distribution** generalization.

ID Generalization We use a RedNet checkpoint trained on the entire SUNRGB-D training split. As these splits were not created with any special biases in mind, the training split should reflect a similar label distribution to the test split.

OOD Generalization We study the setting where the training distribution's $p(y_i, y_j)$ differs from the true distribution's. We do this by picking two object labels that commonly occur together (i.e. picking y_i and y_j such that $p(y_i, y_j)$ is high), and removing all images from the training set where they *do* occur together (thus making $p(y_i, y_j)$ close to zero in the training set). In particular, we choose bed and nightstand as these two objects, and hold out all images in the training set where nightstands and beds co-occur (keeping all other images). After training on this set, we evaluate on the original test split where beds and nightstands frequently co-occur.

3.3. Results

We evaluate the mean intersection-of-union (mIoU) between predicted and ground-truth object segmentations over all object categories for ID and OOD in Table 1.

In each setting, we compare the base model against LAMPP. We see that in both the ID and OOD cases, LAMPP improves upon the baseline image model. The improvements seem small in an absolute sense because we average over 37 object categories. To get a better understanding of the distribution of improvements over object categories, we report *per-category differences in IoU* of our model *relative* to the baseline image model. The rightmost column of Table 1 shows the most-improved and least-improved object categories (and the corresponding IoU change for those categories). We see in both settings that the top object category improved significantly while all other object categories were not significantly affected.

	Model	mIoU	Best/Worst Object (ΔIoU)
ID	Base model	47.8	-
	Model chaining	37.5	shower curtain ($+16.9$)
			toilet (-37.2)
	LAMPP	48.3	shower curtain ($+18.9$)
			desk (-2.16)
OOD	Base model	33.8	-
	LAMPP	34.0	nightstand ($+8.92$)
			sofa (-2.50)

Table 1. Image semantic segmentation results for ID and OOD generalization. We report Intersection-over-Union (IoU) for each model: the base model, a model chaining approach, and LAMPP. We report mIoU (IoUs averaged over each object category), as well as the most- and least-improved object from each method relative to the base model (and the corresponding Δ IoU). LAMPP improves semantic segmentation dramatically on certain categories, while having minimal effect on all other categories.

In the ID setting, the accuracy of detecting *shower curtains* improves by nearly 20 points with LAMPP, as the base model obtains near-0% mIoU on shower curtains, almost always mistaking them for *(window) curtains*. Here, background knowledge from language fixes a major (and previously undescribed) prediction error for a rare class. In the OOD setting, the base image model sees far fewer examples of nightstands and consequently never predicts nightstands on the test data. (*Nightstands* are frequently predicted to be *tables* and *cabinets* instead). This is likewise rectified with LAMPP: background knowledge from language reduces model sensitivity to a systematic bias in dataset construction.

Finally, we see that the model chaining approach repairs prediction errors on the same rare class as LAMPP in the ID setting, but it also introduces new prediction errors on far more classes.

4. LAMPP for Navigation

We next turn to the problem of **object navigation**. Here, we wish to build an agent that, given a goal object g (e.g., a television or a bed), can take actions a to explore and find g in an environment, while using noisy partial observations x

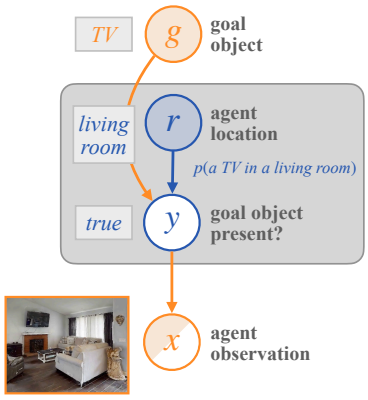

Figure 3. Generative model for **object navigation**. Specifically, given a goal object g, we depict a decomposition of the agent's *success score* y, which takes on value 1 (*true*) if the goal object is present and 0 (*false*) otherwise. We focus on a household domain, where any particular agent location must be within some room r. r then generates the success condition y (indicating whether g is present at the agent location), which generates the agent observation x of the location. Goal objects g and rooms r are given, success conditions y are latent, and agent observations x are partially-observed.

from a camera for object recognition and decision-making. Prior knowledge about where goal objects are likely located can guide this exploration, steering agents away from regions of the environment unlikely to accomplish the agent's goals.

4.1. Methods

Base model We assume access to a pre-trained navigation policy (in this case, from the STUBBORN agent; Luo et al., 2022) that can plan a path to any specified coordinate a in the environment given image observations x. Our goal is to build a *high-level* policy $\pi(a \mid x)$ that can direct this low-level navigation. We focus on navigation in household environments, and assume access to a coarse semantic map of an environment that identifies rooms, but not locations of objects within them. In each state, the STUBBORN low-level navigation policy also outputs a scalar score reflecting its confidence that the goal object is present.

Label space Our high-level policy alternates between performing two kinds of actions a:

- **Navigation**: the agent chooses a room r in the environment to move to. (When a room is selected, we direct the low-level navigation policy to move to a point in the center of the room, and then explore randomly within the room for a fixed number of time steps.)
- **Selection**: whenever an observation is received *during* navigation, the agent evaluates whether it has already reached the goal object. (When the goal object is judged to be present, the episode is ended.)

A rollout of this policy thus consists of a sequence of navigation actions, interleaved with a selection action for every observation obtained while navigating. In both cases, choosing actions effectively requires inference of a specific unobserved property of environment state: whether the goal object is in fact present near the agent. We represent this property with a latent variable y. When navigating, the agent must infer the room that is most likely to contain the goal object. When selecting, the agent must infer whether its current perception is reliable.

We normalize the low-level policy's success score and interpret it as a distribution $p(x \mid y)$, then use the LM to define a distribution $p(y \mid r, g)$. Together, these give a distribution over latent success conditions and observations given goals and agent locations, which may be used to select actions in the high-level policy.

LM queries For $p(y \mid r, g)$, we use the same query as in Section 3 for deriving object–room probabilities, inserting g in place of y_i, except here we do not normalize over object labels (since y is binary), and simply take the relative probability of generating the token *plausible*.

Inference With this model, we define a policy that performs inference about the location of the goal object, then greedily attempts to navigate to the location most likely to contain it. This requires defining $p(a \mid x, g)$ for both navigation and selection steps.

- **Navigation**: the agent chooses a room r maximizing $p(y \mid r, g)$. (The agent does not yet have an observation from the new room, so the optimal policy moves to the room most likely to contain the goal object *a priori*.)
- **Selection**: the agent ends the episode only if $p(y \mid x, r, g) > \tau$ for some confidence threshold τ.

During exploration, the agent maintains a list of previously visited rooms. Navigation steps choose only among rooms that have not yet been visited.

4.2. Experiments

We consider a modified version of the Habitat Challenge ObjectNav task (Yadav et al., 2022). The task objective is to find and move to an instance of the object in unfamiliar household environments as quickly as possible. The agent receives first-person RGBD images, compass readings, and 2D GPS values as inputs at each timestep. In our version of the task, we assume access to a high-level map of the environment which specifies the coordinates and label of each room. Individual objects are not labeled; the agent

must rely on top-down knowledge of where certain objects are likely to be in order to efficiently find the target object.

We implement a MC baseline where the LM guides agent exploration by specifying an ordering of rooms to visit. This is similar to prior work that use LMs to specify high-level policies (Zeng et al., 2023; Sharma et al., 2022), whereby neither LM nor observation model uncertainties are accounted for when generating the *high-level* policy. Details of the MC baseline can be found in Appendix B.1.

We evaluate the ability of the original STUBBORN agent (*base model*), *model chaining*, and our agent (LAMPP) to perform **zero-shot generalization**, where the training data does not contain any information about $p(y \mid r, g)$.[1] We also compare to a *uniform prior baseline* where we preserve the high-level policy of our agent but replace LM priors over object-room co-occurrences with uniform priors:

$$p(y \mid r, g) = \frac{1}{\#\text{ room types in environment}}. \quad (5)$$

Note in the zero-shot case we have no additional information about $p(y \mid r, g)$, so we must assume it is uniform.

4.3. Evaluation & Results

We evaluate success rate (SR), as measured by the percent of instances in which the agent successfully navigated to the goal object. Because the STUBBORN agent is designed to handle only single floors (the mapping module only tracks a 2D map of the current floor), we evaluate only instances in which the goal object is located on the same floor as the agent's starting location.

Results are reported in Table 2. LAMPP far outperforms both the base policy and the policy that assumes uniform priors, in overall and object-wise success rates. We find greatest improvements in goal object categories that have strong tendencies to occur only in specific rooms, such as TV monitors, and less for objects which tend to occur in many different rooms, like plants.

Compared to the MC baseline, LAMPP is better in terms of class-averaged SR, and comparable in terms of frequency-averaged SR. What accounts for this difference in performance? In the MC approach, high-level decisions from the LM and low-level decisions from observation models are usually considered separately and delegated to different phases (it is hard to combine these information sources in string-space): in our implementation, the MC baseline uses the top-down LM for *navigation*, and the bottom-up observation model for *selection*. Because the policy dictated by the LAMPP probabilistic model also ignores bottom-up observation probabilities until the goal object is observed, the *navigation* step of both approaches is functionally equivalent. However, for the *selection* step, we find that combining bottom-up and top-down uncertainties is crucial; in analyses in Appendix B.2, we see that when model uncertainties are ablated, our agent actually *underperforms* a comparable model chaining baseline.

[1] At the time these experiments were conducted, room labels were not yet present in the dataset, so we could only study the zero-shot setting. To evaluate LAMPP, the first two authors of the paper manually annotated room labels in the evaluation set.

Model	Success rate Class	Freq.	Best/Worst Object (ΔSR)
Base model	52.7	53.8	-
Uniform prior	52.1	51.7	-
Model chaining	61.2	65.3	Toilet ($+20.9$) TV Monitor (-4.2)
LAMPP	66.5	65.9	TV Monitor ($+33.0$) Plant (-0.0)

Table 2. Navigation Results for ZS generalization. We report success rates (SR) for the base model, a uniform prior baseline model, a model chaining approach, and LAMPP. We report both a *class-averaged* SR (over goal objects) and a *frequency-averaged* SR (over episodes). We also report the most-improved goal object and least-improved goal object for each method relative to the base model. We find that by using LAMPP, we are able to achieve significant improvement over certain object classes.

Other than performance differences, LAMPP is also substantially more query-efficient: MC requires one query per navigation action of *each episode*, while LAMPP simply requires a fixed number of queries ahead of time, which can be applied to *all actions and episodes*.

5. LAMPP for Action Recognition and Segmentation

The final task we study focuses on video understanding: specifically, taking demonstrative videos of a task (e.g., making an omelet) and segmenting them into actions (e.g., cracking or whisking eggs). Because it is hard to procure segmented and annotated videos, datasets for this task are usually small, and it may be difficult for models trained on task data alone to learn robust models of task-action relationships and action orderings. Large LMs' training data contains much more high-level information about tasks and steps that can be taken to complete them.

5.1. Methods

Base model Given a video of task t, we wish to label each video frame x_i with an action y_i (chosen from a fixed inventory of plausible actions for the task) according to:

$$\arg\max_{y_1 \cdots y_n} p(y_1 \cdots y_n \mid x_1 \cdots x_n, t). \quad (6)$$

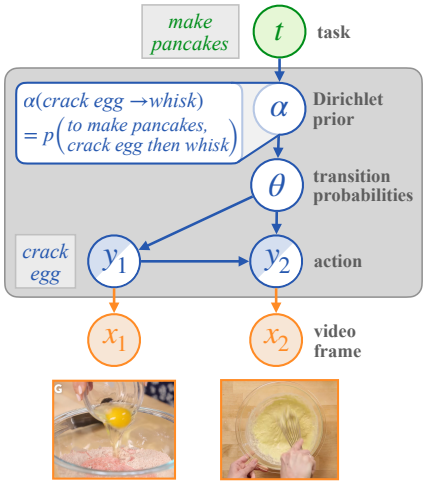

Figure 4. Generative model for **video-action segmentation**. The base model we use for this task is a HMM with transition probabilities parameterized by θ. In this task, we generate a prior over *model parameters* θ: Each task t generates a Dirichlet prior α over action transitions, which in turn generates model parameters θ. θ parameterizes the action transition distribution $y_1 \rightarrow y_2$. Each action y_i at timestep i then generates the observed video frame x_i. Tasks t and video frames x_i are observed, actions y_i are partially-observed, and parameter priors α and parameters θ are latent.

We build on a model by Fried et al. (2020) that frames this as inference in a task-specific hidden Markov model (HMM) in which a latent sequence of actions generates a sequence of video frames according to a distribution:

$$p(x_1, \ldots, x_n \mid y_1, \ldots, y_n) \propto \prod_j p(x_j \mid y_j; \eta)\, p(y_j \mid y_{j-1}; \theta) \tag{7}$$

(omitting the dependence on the task t for clarity). This generative model decomposes into an **emission model** with parameters η and a **transition model** with parameters θ_t, and allows efficient inference of $p(y \mid x)$.[2] $p(y_j \mid y_{j-1}; \theta)$ is a multinomial distribution parameterized by a table of transition probabilities, each of which encodes the probability that action y_{j-1} is followed by action y_j.

In contrast to previous sections, which used pre-trained domain models, here we apply LAMPP to the problem of learning model parameters themselves. Specifically, we use an LM to place a prior on *transition parameters* θ, making it possible to learn about valid action sequences from data while still incorporating prior knowledge from language. Given a dataset of labeled videos of the form $(x_{1\ldots n}, y_{1\ldots n})$, we compute a maximum *a posteriori* estimate of θ:

$$\arg\max_\theta \; \log p(\theta) + \sum_{x,y} \sum_j \log p(y_j \mid y_{j-1}; \theta)\,, \tag{8}$$

(likewise for η). At evaluation time, we use these parameter estimates to label new videos.

Label space We parameterize the prior $p(\theta)$ as a Dirichlet distribution with hyperparameters α, according to which:

$$p(\theta) \propto \prod_i \theta_i^{\alpha_i - 1}\,. \tag{9}$$

Intuitively, the larger α_i is, the more probable the corresponding θ_i is judged to be *a priori*. Here, parameters $\theta_{y \rightarrow y'}$ are probabilities of transitioning from action y to y'; we would like $\alpha_{y \rightarrow y'}$ to be large for plausible transitions, which is achieved by extracting values directly from a LM.

Prompting the LM To derive values of α for each action transition $y \rightarrow y'$, we query the LM with the prompt:

> *Your task is to* [t]. *Here is an *unordered* set of possible actions:* {[Y]}. *Please order these actions for your task. The step after* [y] *can be* [y']

where Y is a set of all available actions for the task. We condition the LM on the non-highlighted portion of the prompt and set $\alpha_{y \rightarrow y'} = \lambda \cdot p_{\text{LM}}(y' \mid \text{prompt}(y))$ (the probability of completing the prompt with the action name y'), where λ controls the strength of the prior.

Inference The use of a Dirichlet prior means that Eq. (8) has a convenient closed-form solution:

$$\theta_{y \rightarrow y'} = \frac{\alpha_{y \rightarrow y'} + \#(y \rightarrow y') - 1}{(\sum_{y''} \alpha_{y \rightarrow y''}) + \#(y) - |Y|}\,, \tag{10}$$

where $\#(y \rightarrow y')$ denotes the number of occurrences of the transition $y \rightarrow y'$ in the training data, $\#(y)$ denotes the number of occurrences of y in the training data, and $|Y|$ is the total number of actions.

5.2. Experiments

We evaluate using the CrossTask dataset (Zhukov et al., 2019), which features instructional videos depicting tasks (e.g., *make pancakes*). The learning problem is to segment videos into regions and annotate each region with the corresponding action being depicted (e.g., *add egg*).

We evaluate the ability of the *base model* and LAMPP to perform **zero-shot** and **out-of-distribution** generalization. For all experiments with LAMPP, we use $\lambda = 10$. We do not study a MC baseline for this task, as model chaining is unable to generate parameters rather than labels.

[2] Fried et al. (2020)'s model is a hidden *semi*-Markov model (HSMM) in which latent action states generate multiple lower-level actions in sequence. While our experiments also use an HSMM, we omit the HSMM emission model for clarity of presentation.

		Recall (class avg.)	Recall (freq avg.)
ZS	Base model	44.4	46.0
	LAMPP	45.7	47.9
OOD	Base model	37.6	40.9
	LAMPP	38.1	41.2

Table 3. Video segmentation results for ZS and OOD generalization. We report step recall for the base model and LAMPP. We report both a *class-averaged* step recall (over goal objects) and a *frequency-averaged* step recall (over videos). We also report the most-improved action and least-improved action for LAMPP relative to the base model in each setting. LAMPP provides a significant improvement in certain task classes.

ZS Generalization We assume that the training data contains no information about the transition distribution $p(y_i \mid y_{i-1}, t)$. However, we still assume access to *all video scenes and their action labels*, which allows us to learn emission distributions $p(x_i \mid y_i)$. We do this by assuming access to only an *unordered set* of video frames from each task, where each frame is annotated with its action label, but with no sense of which frame preceded or followed it.

Because we have no access to empirical counts of transitions from the training data, the model falls back completely on its priors when computing those parameters:

$$\theta_{y \to y'} = \frac{\alpha_{y \to y'} - 1}{(\sum_{y''} \alpha_{y \to y''}) - |Y|}$$

which is uniform for the base model and derived from the LM for LAMPP.

OOD Generalization We bias the *transition distribution* by randomly sampling a common transition from each task and holding out all videos from the training set that contain that transition.

5.3. Evaluation & Results

Following Fried et al. (2020), we evaluate *step recall*, i.e. the percentage of actions in the real action sequence that are also in the model-predicted action sequence. For simplicity, we ignore background actions during evaluation.

Results are shown in Table 3. For both the ZS and OOD settings, step recall slightly improves with LAMPP. The small magnitude of improvement may be because the LM sometimes does not possess a sensible prior over action sequences (compared to room–object co-occurrences, which it possesses accurate and calibrated priors for). For example, it is heavily biased towards returning actions in the order they are presented in the prompt.[3]

[3]We tried over 20 prompts, verifying whether the predicted action order looked sensible, but all yielded mixed results. We used the best prompt for these experiments.

Indeed, the transitions that see most improvement with LAMPP are also the ones for which LM priors are more aligned with the test data than the training-set priors. For example, in the OOD setting, the held-out transitions' recalls improve by an average of **8.2%**.

6. Related Work

String Space Model Chaining There has been much recent work in combining and composing the functionality of various models *entirely in string space*. The Socratic models framework (Zeng et al., 2023) proposes chaining together models operating over different modalities by converting outputs from each into natural language strings. Inter-model interactions are then performed purely in natural language.

While such methods have yielded good results in many tasks, like egocentric perception and robot manipulation (Ahn et al., 2022), they are fundamentally limited by the expressivity of the string-valued interface. Models often output useful features that cannot be easily expressed in language, such as graded or probabilistic uncertainty (e.g., in a traditional image classifier). Even if such information is written in string form, there is no guarantee that language models will correctly use it for formal symbolic reasoning–today's largest LMs still struggle with arithmetic tasks expressed as string-valued prompts (Ye & Durrett, 2022).

Concurrent to the present work is the approach of Choi et al. (2022), which similarly seeks to use language model scores as a source of common-sense information in other decision-making tasks. There, LMs are applied to feature selection, reward shaping, and casual inference tasks, rather than used to provide explicit priors for probabilistic models.

LMs and Probabilistic Graphical Models Interpretation of LMs as composable probability distributions is well studied in pure language-processing tasks. Methods like chain-of-thought question-answering (Wei et al., 2022), thought verification (Cobbe et al., 2021), and bootstrapped rationale-generation (Zelikman et al., 2022) may all be interpreted as probabilistic programs encoded as repeated language model queries (Dohan et al., 2022). However, this analysis exclusively considers language tasks; to the best of our knowledge, the present work is the first to specifically connect language model evaluations to probabilistic graphical models in non-language domains.

7. Conclusion

We have described LAMPP, a generic technique for integrating background knowledge from language into decision-making problems by extracting probabilistic *priors* from language models. LAMPP improves zero-shot, out-of-

distribution, and in-distribution generalization across image segmentation, household navigation, and video-action recognition tasks. It enables principled composition of uncertain perception and noisy common-sense and domain priors, and shows that language models' comparatively unstructured knowledge can be integrated naturally into structured probabilistic approaches for learning or inference. The effectiveness of LAMPP depends crucially on the quality of the LMs used to generate priors. While remarkably effective, today's LMs still struggle to produce calibrated plausibility judgments for some rare tasks. Improving LM knowledge representations is an important problem not just for LAMPP but across natural language processing; as the quality of LMs for core NLP tasks improves, we expect that their usefulness for LAMPP will improve as well.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Nos. 2238240 and 2212310. BZL is supported by a NDSEG Fellowship. We would like to thank Luca Carlone for valuable discussions regarding the design of navigation experiments.

References

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

Ammanabrolu, P., Cheung, W., Broniec, W., and Riedl, M. O. Automated storytelling via causal, commonsense plot ordering. In *AAAI Conference on Artificial Intelligence*, 2020.

Ammanabrolu, P., Urbanek, J., Li, M., Szlam, A., Rocktäschel, T., and Weston, J. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 807–833, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.64. URL https://aclanthology.org/2021.naacl-main.64.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Choi, K., Cundy, C., Srivastava, S., and Ermon, S. LMPriors: Pre-trained language models as task-specific priors. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022. URL https://openreview.net/forum?id=U2MnmJ7Sa4.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021.

Dohan, D., Xu, W., Lewkowycz, A., Austin, J., Bieber, D., Lopes, R. G., Wu, Y., Michalewski, H., Saurous, R. A., Sohl-dickstein, J., Murphy, K., and Sutton, C. Language model cascades. In *International Conference on Machine Learning*, 2022.

Fried, D., Alayrac, J.-B., Blunsom, P., Dyer, C., Clark, S., and Nematzadeh, A. Learning to segment actions from observation and narration. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2569–2588, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.231. URL https://aclanthology.org/2020.acl-main.231.

Jiang, J., Zheng, L., Luo, F., and Zhang, Z. Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation, 2018. URL https://arxiv.org/abs/1806.01054.

Kveraga, K., Ghuman, A., and Bar, M. Top-down predictions in the cognitive brain. *Brain and Cognition*, 65(2): 145–168, 2007.

Lew, A. K., Tessler, M. H., Mansinghka, V. K., and Tenenbaum, J. B. Leveraging unstructured statistical knowledge in a probabilistic language of thought. *Proceedings of the Annual Conference of the Cognitive Science Society*, 2020.

Li, B. Z., Nye, M., and Andreas, J. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1813–1827, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.143. URL https://aclanthology.org/2021.acl-long.143.

Luo, H., Yue, A., Hong, Z.-W., and Agrawal, P. Stubborn: A strong baseline for indoor object navigation, 2022.

Mirault, J., Snell, J., and Grainger, J. You that read wrong again! a transposed-word effect in grammaticality judgments. *Psychological Science*, 29:095679761880629, 10 2018. doi: 10.1177/0956797618806296.

Painter, C. *Learning Through Language in Early Childhood*. Continuum Collection. Bloomsbury Publishing, 2005. ISBN 9781847143945. URL https://books.google.com/books?id=4sB0i-DfT0MC.

Sharma, P., Torralba, A., and Andreas, J. Skill induction and planning with latent language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1713–1726, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.120. URL https://aclanthology.org/2022.acl-long.120.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. Progprompt: Generating situated robot task plans using large language models. In *Second Workshop on Language and Reinforcement Learning*, 2022. URL https://openreview.net/forum?id=aflRdmGOhw1.

Song, S., Lichtenberg, S., and Xiao, J. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 567–576. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298655. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298655.

Talmor, A., Yoran, O., Bras, R. L., Bhagavatula, C., Goldberg, Y., Choi, Y., and Berant, J. CommonsenseQA 2.0: Exposing the limits of AI through gamification. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=qF7FlUT5dxa.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Yadav, K., Ramakrishnan, S. K., Turner, J., Gokaslan, A., Maksymets, O., Jain, R., Ramrakhya, R., Chang, A. X., Clegg, A., Savva, M., Undersander, E., Chaplot, D. S., and Batra, D. Habitat challenge 2022. https://aihabitat.org/challenge/2022/, 2022.

Ye, X. and Durrett, G. The unreliability of explanations in few-shot prompting for textual reasoning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. STar: Bootstrapping reasoning with reasoning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Zeng, A., Attarian, M., Ichter, B., Choromanski, K., Wong, A., Welker, S., Tombari, F., Purohit, A., Ryoo, M., Sindhwani, V., Lee, J., Vanhoucke, V., and Florence, P. Socratic models: Composing zero-shot multimodal reasoning with language. In *Submitted to The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=G2Q2Mh3avow. under review.

Zhukov, D., Alayrac, J.-B., Cinbis, R. G., Fouhey, D., Laptev, I., and Sivic, J. Cross-task weakly supervised learning from instructional videos. In *Computer Vision and Pattern Recognition*, 2019.

A. LAMPP for Semantic Segmentation

A.1. Methods

We derive the following decision rule from the model in Fig. 2:

$$p(y_i \mid \underline{x}) \propto\!\!\!\!\sim \; p(y_i \mid d_i = d_i^*) p(d_i = d_i^* \mid x_i) \left(\sum_r p(r) p(y_i \mid r) \prod_{j=1 \cdots n} \left(\sum_{y_j} \frac{p(r \mid y_j) p(d_j = y_j \mid x_j)}{p(r)} \right) \right) \tag{11}$$

We obtain this decision rule as described below. Here we denote rooms r, true object labels y, noisy object labels d, and observations x. (Underlines denote sets of variables, so e.g., $\underline{x} = \{x_1, \ldots, x_n\}$.) Finally, we write d_i^* to denote the base model's prediction for each image segment ($d_i^* = \arg\max p_{\text{seg}}(d_i \mid x_i)$). To see this:

$$\begin{aligned}
p(y_i \mid \underline{x}) &\propto \sum_r \sum_{\underline{y} \setminus \{y_i\}} \sum_{\underline{d}} p(\underline{x}, \underline{y}, \underline{d}, r) \\
&= \sum_r \sum_{\underline{y} \setminus \{y_i\}} \sum_{\underline{d}} p(r) \Big(p(y_i \mid r) p(d_i \mid y_i) p(x_i \mid d_i) \Big) \prod_j p(y_j \mid r) p(d_j \mid y_j) p(x_j \mid d_j) \\
&= \sum_r p(r) \Big(p(y_i \mid r) \sum_{d_i} p(d_i \mid y_i) p(x_i \mid d_i) \Big) \Big(\prod_j \sum_{y_j} p(y_j \mid r) \sum_{d_j} p(d_j \mid y_j) p(x_j \mid d_j) \Big)
\end{aligned}$$

Rather than marginalizing over all choices of d, we restrict each sum to a single term. For d_i, we choose the most likely detector output $d_i = d_i^*$. For d_j, we choose the corresponding y_j in the outer sum. Together, these simplifications reduce the total number of unnecessary LM queries about unlikely object confusions, and give a lower bound:

$$\geq p(d_i = d_i^* \mid y_i) p(x \mid d_i = d_i^*) \sum_r p(r) (p(y_i \mid r) \Big(\prod_j \sum_{y_j} p(y_j \mid r) p(d_j = y_j \mid y_j) p(x_j \mid y_j) \Big)$$

Applying Bayes' rule locally:

$$\begin{aligned}
&= \frac{p(y_i \mid d_i = d_i^*) p(d_i = d_i^*)}{p(y_i)} \frac{p(d_i = d_i^* \mid x_i) p(x_i)}{p(d_i = d_i^*)} \\
&\quad \sum_r p(r) p(y_i \mid r) \Big(\prod_i \sum_{y_j} \frac{p(r \mid y_j) p(y_j)}{p(r)} p(d_j = y_j \mid y_j) \frac{p(d_j = y_j \mid x_j) p(x_j)}{p(d_j = y_j)} \Big)
\end{aligned}$$

Finally, we make two modeling assumptions. First, we assume that of the form $p(y)$ and $p(d)$—the marginal distributions of true and noisy object labels—are uniform. This allows us to use LMs as a source of information about object co-occurrence probabilities without relying on their assumptions about base class frequency. Second, for non-target detections x_j, we assume the probability that noisy labels match the true labels is constant over object categories. Then, dropping constant terms gives:

$$\propto p(y_i \mid d_i = d_i^*) p(d_i = d_i^* \mid x_i) \sum_r p(r) p(y_i \mid r) \Big(\prod_j \sum_{y_j} \frac{p(r \mid y_j)}{p(r)} p(d_j = y_j \mid x_j) \Big)$$

A.2. Model Chaining Baseline

The model chaining baseline is given model predictions $\widehat{d}_i$ for *each segment* x_i and re-labels each segment by querying GPT-3 with:

You can see: [$\underline{\widehat{d}}$]

Model	Class-Avg. SR	Freq.-Avg. SR
LAMPP	66.5	65.9
$-p(y \mid r)$ during selection	58.8	64.9
Model chaining	61.2	65.3

Table 4. Navigation results verification ablations. We ablate the LM uncertainties over $p(y \mid r)$ when computing the selection action, making LAMPP functionally similar to a model chaining baseline. We find that having these uncertainties are crucial; without them, LAMPP actually *underperforms* the model chaining baseline.

You are in the [r]
The thing that looks like [$\widehat{d}_i$] *is actually* [y_i].

The LM is given the non-highlighted portions and asked to generate the portions highlighted in yellow. $\underline{\widehat{d}}$ is the set of all unique objects detected by the base model, written out as a comma-separated list. r is a room type generated by the LM based on these objects (inferred by normalizing over possible room types), and y_i is the actual identity of the object corresponding to this segment. We replace all pixels formerly predicted as $\widehat{d}_i$ with y_i.

B. LAMPP for Navigation

B.1. Model Chaining Baseline

As in the image segmentation case, we have a model chaining baseline. LM priors are integrated into exploration through directly querying the LM with

The house has: [$\underline{r}$].
You want to find a [g]. *First, go to each* [r_0]. *If not found, go to each* [r_1]. *If not found, go to each* $\cdots$

whereby $\underline{r}$ is a list of all room types in the environment, for example, *3 bathrooms, 1 living room, 1 bedroom*. The LM returns the best room type r_0 to navigate to in order to find g. The agent visits all r_0 in order of proximity. If the object is not found, the LM is queried for the next best room type to visit, etc., until the object is found or we run out of rooms in the environment.

B.2. Additional Analysis

Why does LAMPP outperform model chaining? As noted in Section 4.3, model chaining approaches do not use bottom-up observational probabilities or top-down LM probabilities when generating their high-level policy. Our method does, specifically integrating both probabilities when performing selection (recall we threshold $p(y \mid x, r, g)$ at selection steps, which decomposes to $p(y \mid x)p(y \mid r, g)$). The model chaining equivalent to this phase simply delegates selection to the low-level model, which only uses observational uncertainties $p(y \mid x)$.

To further understand and how using LM probabilities contributes at this phase, we run a version of LAMPP where we simply change the decision rule at the selection action to $p(y \mid x)$. Results are reported in Table 4. Note that we actually *underperform* the MC baseline when we take away top-down uncertainties $p(y \mid r, g)$ — once again highlighting the importance of combining both sources of uncertainty.